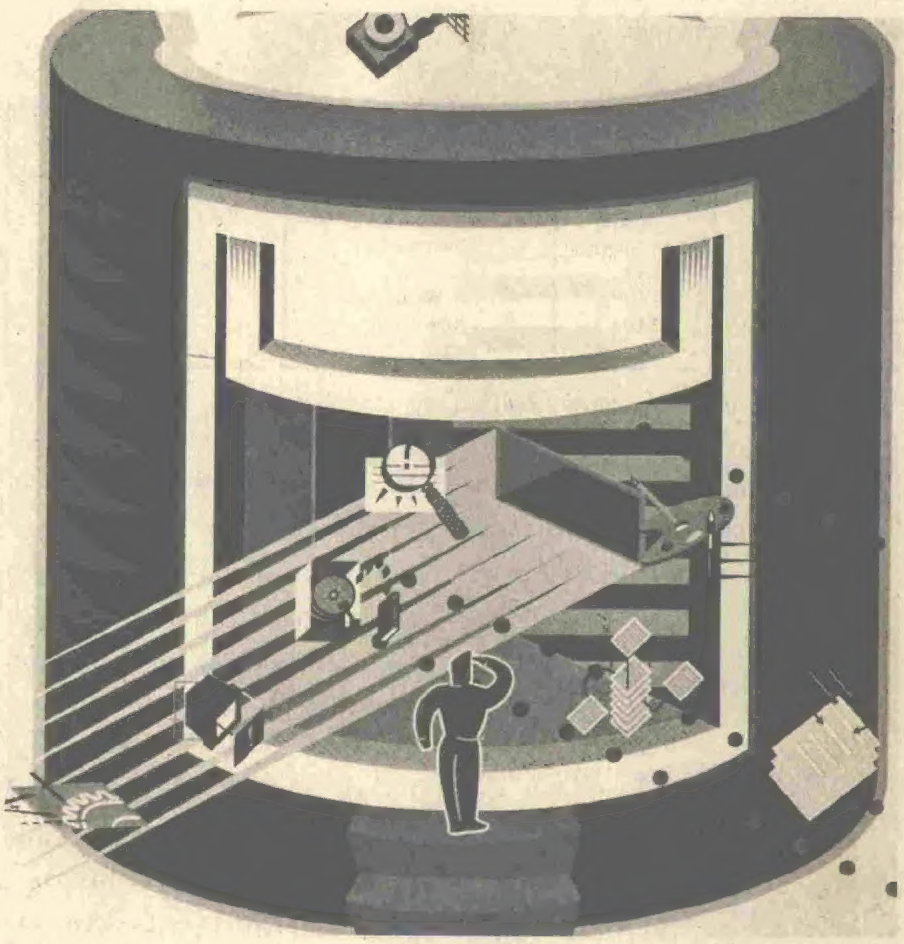
พัลลภ เฉลิมฉัตร

ร่ายยาว GUIs:
สิ่งที่ควรรู้เกี่ยวกับวินโดว์ (1)

ศึกษาความวิจิตรพิสดาร ในการทำงานด้านกราฟิก มัลติมีเดียสเซอร์และฮาร์ดแวร์ ที่กำลังมาแรงในขณะนี้



ปัจจุบันเรามักพบเห็นความก้าวหน้าทางด้านซอฟต์แวร์คอมพิวเตอร์กันมาก ในแต่ละสาขาจะมีความวิจิตรพิสดารมากขึ้นเรื่อย ๆ และตัวที่ถูกกล่าวขวัญถึงและได้รับความสนใจมากที่สุดในช่วงนี้ก็คือไมโครซอฟต์วินโดว์ 3.0 ของบริษัทไมโครซอฟต์ และในครั้งนี้ทางบริษัทได้ทุ่มเททรัพยากรจำนวนมหาศาลทั้งในการพัฒนาและการจัดงานแถลงเปิดตัว จนประสบความสำเร็จมากพอสมควรทีเดียว แต่มีหลาย ๆ คนกล่าวว่าวินโดว์ 3.0 จะเป็นแค่ซอฟต์แวร์ซินเดอเรลล่า (คำเปรียบเปรยในที่นี้หมายความว่าวินโดว์ 3.0 เป็นซอฟต์แวร์ที่ดังมากและมียอดขายสูงในช่วงที่ผ่านมา ทั้งนี้อาจเป็นเพราะความอยากลองเล่นของบรรดาชาวคอมพิวเตอร์ แต่เมื่อต้องการทำงานจริง ๆ จะมีสักกี่คนที่ใช้ความสามารถของวินโดว์ในการทำงาน (Multisingle tasking) โดยเฉพาะในบ้านเรา) และคงมีหลาย ๆ ท่านที่สนใจการเขียนโปรแกรมแบบวินโดว์รวมทั้งตัวผมด้วย การเขียนโปรแกรมแบบวินโดว์ในที่นี้ไม่ได้หมายถึงไมโครซอฟต์วินโดว์เท่านั้น (ต่อไปจะขอเรียกย่อ ๆ ว่าวินโดว์) แต่หมายถึงระบบวินโดว์อื่น ๆ ด้วย เช่น X Window, Motif เป็นต้น ซึ่งในฐานะที่ผมได้คลุกคลีกับบรรดา GUIs ต่าง ๆ คิดว่าพอจะมีเรื่องราวน่าสนใจมาเล่าให้ทุกท่านได้ทราบกันถ้วนหน้า

ในตอนต้นนี้จะขอเน้นแนวความคิดหลักของระบบวินโดว์ โดยอ้างอิงกับวินโดว์ 3.0 เพราะเป็นระบบวินโดว์ที่สามารถหาง่ายที่สุดในบ้านเรา ตารางที่ 1 จะเป็นคำจำกัดความที่ใช้ในบทความ

GUIs ไม่ใช่ Window System

คำว่า GUIs หรือ Graphical User Interface System ที่เราเคยได้ยินกันบ่อย ๆ นั้นมีความหมายไม่เหมือนกับ

คำ	ความหมาย
ระบบวินโดว์	หมายถึงซอฟต์แวร์ที่ใช้แนวความคิดเกี่ยวกับวินโดว์ เช่น Motif, X Window, Open Loop, Microsoft Windows
Window	หมายถึงกรอบสี่เหลี่ยมที่เปิดโดยระบบวินโดว์
Windows	หมายถึงซอฟต์แวร์ระบบวินโดว์ของบริษัทไมโครซอฟต์ชื่อ Microsoft Windows

ตารางที่ 1 คำจำกัดความที่ใช้ร่วมกัน

GUIs	API	Window System	Imaging Model
Windows	GDI	Graphics Device Interface	GDI Output functions
PM	User Interface API	Windows API	Graphics API
Motif	XUI	X Window	***
Open Desktop	XUI	X Window	***

ตารางที่ 2 ตัวอย่าง GUIs (ที่มี *** หมายความว่ายังไม่มีข้อมูล)

วินโดว์ซิสเต็มส์ (Window system) หรือระบบวินโดว์เลยทีเดียว แต่มีนิยามที่ทางแมคอินทอชกล่าวไว้สำหรับ GUIs ดังนี้

– มี pointing device ซึ่งปกติคือเมาส์

– ทุกเมนูบนจอภาพอยู่ภายใต้การควบคุมของ pointing device

– วินโดว์ต่าง ๆ จะแสดงสิ่งที่คอมพิวเตอร์กำลังทำ

– มีไอคอน (icon) ใช้แทนไฟล์, ไดเรคทอรีและอื่น ๆ

– มี dialog boxes, buttons, check boxes และ graphical Widgets เพื่อให้ผู้ใช้บอกคอมพิวเตอร์ว่าจะทำอะไรอย่างไร

ปัจจุบัน GUIs บางตัวมีไม่ครบทุกอย่างที่กล่าวมา เช่น บางระบบไม่มีการใช้ไอคอน ส่วนใหญ่ GUIs จะมีลักษณะไปในทางเดียวกันเมื่อมองการใช้งานบนจอภาพ แต่ภายในยังแตกต่างกัน ปกติแล้ว GUIs มักประกอบด้วย 3 ส่วนสำคัญคือ

1. วินโดว์ซิสเต็มส์ ซึ่งเป็นเซตของโปรแกรมมิ่งทูลส์และคำสั่ง windows, menus, dialog boxes ใช้ควบคุมการสร้างหน้าต่าง การย้ายหน้าต่าง เป็นต้น

X Window นั้นเป็นแค่วินโดว์ซิสเต็มส์เท่านั้นไม่ได้เป็น GUIs สาเหตุที่ถูกใช้เป็นวินโดว์ซิสเต็มส์ในหลาย ๆ GUIs เช่น Motif, Open Desktop ก็เลยทำให้บรรดา GUIS เหล่านั้นสามารถจะขอแชร์ (share) โปรแกรมมิ่งทูลส์สำหรับทำแอพพลิเคชั่นโปรแกรมร่วมกันได้ และนี้ก็เป็นเหตุบางประการที่ทำให้ Windows มีชื่อเสียงเป็นที่รู้จักกันทั่วไป

2. Imaging model จะกำหนดรูปแบบในการสร้างฟอนต์และกราฟิกบนจอภาพ ตัวอย่างที่เห็นได้ชัดคือโพสต์-คริปต์เป็น imaging model

3. แอพพลิเคชั่นโปรแกรมอินเตอร์-เฟส (Application program interface: API) เป็นฟังก์ชันที่โปรแกรมเมอร์ใช้ในการระบุ windows, menus, scroll bars, icons ที่แสดงบนจอ

ในตารางที่ 2 แสดง GUIs บางชนิดพร้อมส่วนต่าง ๆ ที่มีให้ลองเปรียบเทียบกัน

ลักษณะเด่นของวินโดว์

สำหรับวินโดว์และ GUIs โดยทั่วไปมักมีคุณสมบัติสำคัญที่น่าสนใจต่อโปรแกรมเมอร์อยู่ 3 อย่างคือ

1. Graphics-oriented User Interface

2. Multitasking (สำหรับ GUIs ที่ทำ Multitask ได้)

3. Hardware Independence

Graphics-oriented User Interface

สิ่งนี้เป็นจุดเด่นสำคัญที่สุดสำหรับผู้ใช้โดยเฉลี่ย เนื่องจากวินโดว์จะใช้รูปภาพแทนความหมายต่าง ๆ เช่น รูไดรฟ์ใน File Manager เป็นต้น มีการใเมนูกับ pointing device (ในวินโดว์คือเมาส์ แต่ในระบบอื่นอาจใช้อย่างอื่นแทเมาส์ก็ได้) ร่วมกับการใช้คีย์บางคีย์แทการใช้คำสั่งยาว ๆ เราอาจจะคิดเสมือว่ารูปภาพและกราฟิกเหล่านี้เป็น enhanc text information ก็ได้ เราจะใช้คำว Graphics object เรียก menu, scroll ba นอกจากนี้ยังมี Graphics Object พวหนึ่งที่ใช้ในการสร้าง Graphics Obje ด้วยกันเอง ได้แก่ pens, fonts, brushe ต่าง ๆ ในวินโดว์เราสามารถที่จะเตรีย external file (นิยมเรียกว่า resourc file) ซึ่งเป็นเท็กซ์ไฟล์ล้วน ๆ อธิบายถึ Graphics Object ที่เราประสงค์ให้ เช่น เราต้องการ dialog box ที่มี te entry box อยู่ด้วย วินโดว์จะนำข้อมูลไปทำการสร้าง dialog box โดยโปรแกรของเราสามารถสั่งให้วินโดว์แสดง dialo

x นี้เมื่อไรก็ได้ตามความต้องการในโปร-
รมของเรา สำหรับ Graphics objects
วินโดว์เตรียมไว้แล้วเรียกว่า Stock
bject ช่วยให้ประหยัดเวลาในการ
ฒนา Graphics Objects ด้วยตัวเอง

ในบางกรณีถ้าต้องการพัฒนา Gra-
ics Objects ขึ้นเองเราสามารถทำได้
สาเหตุอาจเป็นเพราะ

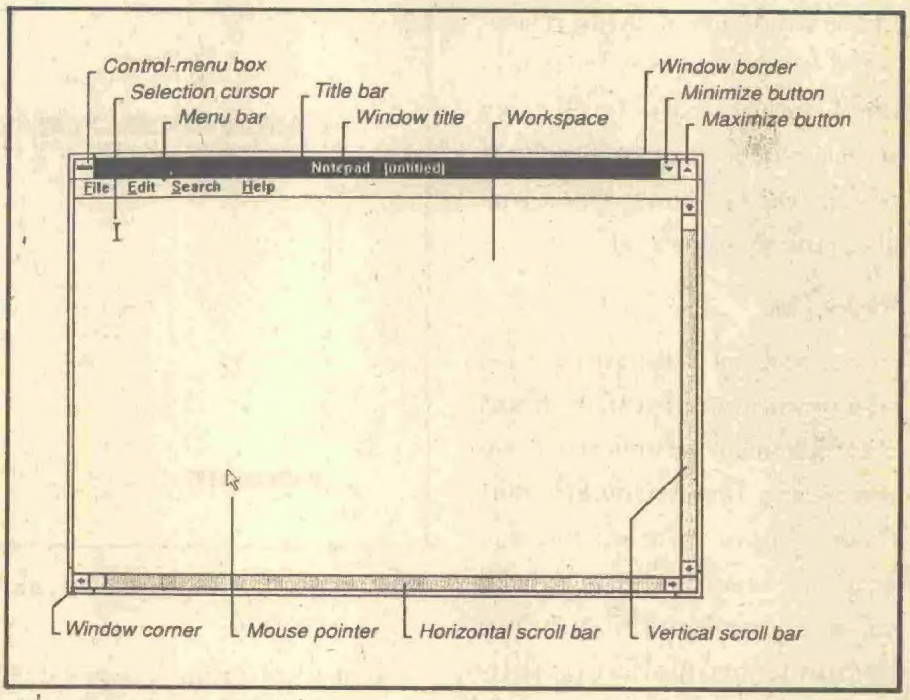
1. Graphics Objects นั้นไม่มี
ระบบ

2. ต้องการแก้ไข Graphics Ob-
ts ที่มีไว้ให้ เราสามารถเรียกบรรดา Gra-
ics objects ทั้งหลายว่าเป็น resou
e ได้เช่นเดียวกับ resource ที่มีอยู่
ระบบ เช่น หน่วยความจำ ซีพียู

หัวใจสำคัญของวินโดว์อินเตอร์เฟส
อ Graphics Device Interface หรือ
DI โดย GDI นี้ก็เป็นไลบรารีของรูทีน
กี่ยวข้องกับการอินเตอร์เฟส ซึ่งทั้ง
วินโดว์และแอพพลิเคชั่นสามารถ
งานร่วมกับ GDI ได้ ปัญหาที่ตามมา
งานอินเตอร์เฟสส่วนนั้นใครเป็นคน
วินโดว์หรือแอพพลิเคชั่น เช่น การเปิด
log box ในโปรแกรม วินโดว์จะเป็น
าดและจัดการเรื่อง user entry และปิด
log box เมื่อผู้ใช้จบการทำงาน แต่
้าที่ของโปรแกรมเราจะจัดการวาด
มูลภายในวินโดว์ใหม่แทนที่บริเวณ
ยเปิด dialog box

อดีของการเขียนโปรแกรม
นวินโดว์

โดยส่วนใหญ่คนที่เขียนโปรแกรม
ระบบวินโดว์จะต้องมีความเข้าใจ
ระบบวินโดว์และ Kit routines ส่วน
ดีของการเขียนโปรแกรมบนวินโดว์
สรุปได้ 4 ประการคือ



รูปที่ 1 แสดงส่วนประกอบของวินโดว์

1. วินโดว์อินเตอร์เฟสช่วยให้การ presentation ข้อมูลข่าวสารเด่นชัดขึ้น เนื่องจากการแสดงผลกราฟิก สามารถแสดงข้อความและตัวเลขสถิติกำกับบนกราฟได้

2. การใช้งานโปรแกรมบนวินโดว์จะมีลักษณะคล้ายคลึงกัน เช่น มีเมนูการใช้แบบเดียวกันไม่ว่าจะเป็นบนโปรแกรมใด ๆ ก็ตาม (Program Consistency)

3. ช่วยลดภาระของโปรแกรมเมอร์ในการทำยูสเซอร์อินเตอร์เฟสเอง

4. Hardware Independence เช่น รูทีนเดียวกันใช้พิกัด x, y เดียวกัน สามารถใช้ได้กับหลาย ๆ ดีไวซ์ โปรแกรมเมอร์ไม่ต้องห่วงว่าโปรแกรมนี้จะใช้บนดีไวซ์ตัวไหน เพราะหน้าที่นี้วินโดว์จะจัดการเสร็จสรรพ

พื้นที่บนหน้าต่างกับไอคอน

เมื่อเรามองจอภาพจะเห็นว่าวินโดว์ได้แบ่งจอภาพออกเป็น 2 ส่วนคือส่วนบน เรียกว่า Window area เป็นส่วนที่กินเนื้อที่บนจอเกือบทั้งหมด โดยแสดงโปรแกรมต่าง ๆ ที่ต้องการความสนใจในขณะนั้น อีกส่วนหนึ่งเรียกว่า icon area ส่วนนี้จะอยู่ทางด้านล่าง มีบริเวณเล็ก ๆ เก็บโปรแกรมที่ยังไม่ต้องการความสนใจมากนักหรืออาจเป็นงานบางอย่าง เช่น print spooler อาทิ โปรแกรมนัดหมายงานอาจอยู่ในรูปไอคอนเมื่อมีโทรศัพท์ติดต่อเข้ามา โปรแกรมนี้จะเริ่มทำงานทันที

ส่วนประกอบของแต่ละวินโดว์

โดยทั่วไปหน้าต่างทุก ๆ บานมักมีส่วนประกอบคล้ายกัน ดังในรูปที่ 1

Child window

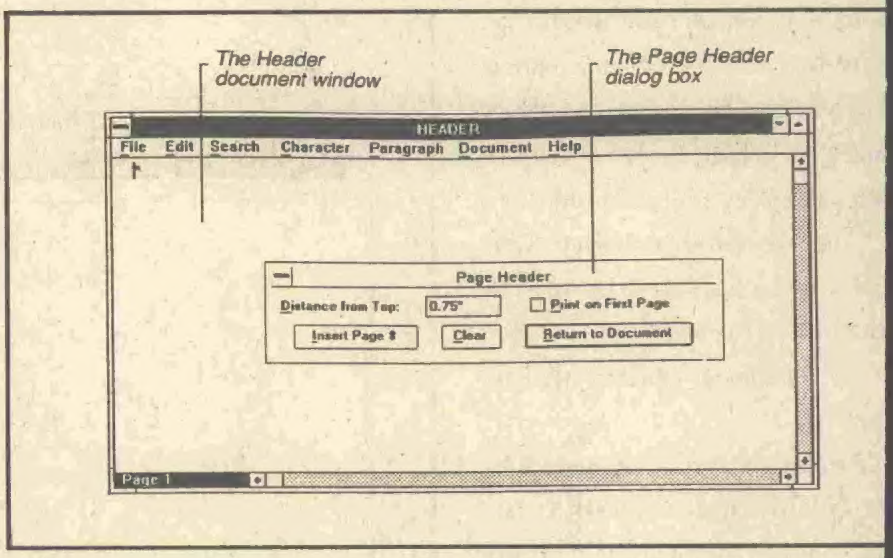
บางครั้งโปรแกรมของเราอาจจะต้องทำงานหลายอย่าง ดังนั้นเราสามารถใช้วิธีเปิดหน้าต่างสำหรับแต่ละงานแยก

กันภายในตัวโปรแกรมเดียวกัน เราเรียกหน้าต่างที่เปิดเพื่อทำงานใดงานหนึ่งเฉพาะว่า Child window ดังรูปที่ 2 โดยที่ Child Window สามารถเคลื่อนย้ายได้ แต่จะถูก clip กับขอบของ Parent Window (หน้าต่างผู้ปกครอง)

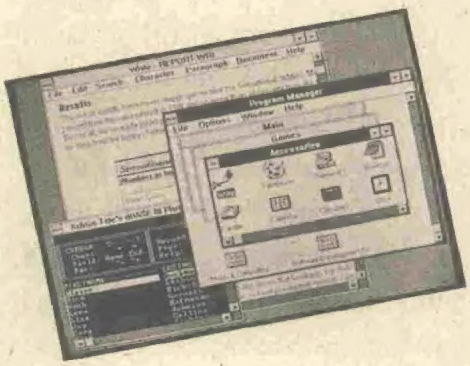
Window Class

บ่อยครั้งที่โปรแกรมที่เราสร้างหรือเคยพบเห็นจะสร้างหน้าต่างขึ้นมา 2 อันที่มีลักษณะทั้งหน้าตาและการทำงานคล้ายคลึงกัน โดยเฉพาะเมื่อมีเรื่องของ Child Window เข้ามาเกี่ยวข้องด้วย การปรากฏของหน้าต่างหลาย ๆ อันที่มีหน้าตาและการทำงานเหมือนกันนี้พบบ่อยมากเมื่อมีการเปิดใช้หน้าต่างโปรแกรมมากกว่าหนึ่งขึ้นไป เช่นโปรแกรมไมโครซอฟต์ยอมให้เราก็อบปี้ข้อมูลจากไฟล์หนึ่งไปอีกไฟล์หนึ่งได้โดยมีอินสแตนซ์ของ Microsoft Paint อยู่ 2 อัน โดยตัดข้อมูลจากอินสแตนซ์หนึ่งแล้วส่งต่อให้อีกอินสแตนซ์หนึ่ง ในแต่ละอินสแตนซ์ของโปรแกรมที่ใช้งานจะสร้างวินโดว์ของมันเอง พร้อมกับดาต้าเซกเมนต์แต่จะใช้โค้ดร่วมกันบางส่วน เราเรียกบรรดาหน้าต่างที่มีพฤติกรรมคล้าย ๆ กันและหน้าตาเหมือนกันว่าอยู่ในคลาส (Class) เดียวกัน โดยทุก ๆ หน้าต่างจะต้องเป็นหน้าต่างที่ถูกระบุคลาสแล้ว (a window of some predefined window class) มีคลาสบางอย่างที่ถูกระบุไว้เรียบร้อยแล้วกับตัววินโดว์

การแบ่งหน้าต่างเป็นคลาสจะช่วยให้เราไม่ต้องจัดการเกี่ยวกับรายละเอียดของข้อมูลประจำตัวของแต่ละหน้าต่างมากจนเกินไป ทั้งนี้เพราะแต่ละวินโดว์คลาสมีโครงสร้างข้อมูลสำหรับเก็บรายละเอียดส่วนตัว เช่น รูปแบบเคอร์เซอร์เป็นต้น หน้าต่างอื่น ๆ ที่อยู่ในคลาสจะใช้ข้อมูลประจำตัวร่วมกัน นอกจากที่สำคัญที่สุดคือ ทุกวินโดว์คลาสจะมีวินโดว์โพรซีเดอร์ (Window Procedure) ของตัวเอง ดังนั้นหน้าต่างทั้งสองที่อยู่ในคลาสเดียวกัน จะใช้โค้ดส่วนนี้ร่วมกันเพื่อประหยัดพื้นที่ในหน่วยความจำ โดยแต่ละตัวจะมีดาต้าเซกเมนต์แยกจากกัน เช่น Title Bar ก็เป็น Window Class หนึ่ง ซึ่งมีวินโดว์โพรซีเดอร์ระบุหน้าที่การทำงานของมัน เมื่อถูกผู้ใช้ลากไปมาบนจอ แต่มันก็มีข้อมูลประจำตัวของมันแยกจาก Title Bar อื่น ๆ เช่น ชื่อของโปรแกรมที่อยู่บน Title Bar รายละเอียดเรื่องอินสแตนซ์จะกล่าวละเอียดอีกครั้งหนึ่งในหัวข้อมัลติทาสกิ้ง



รูปที่ 2 Page Header เป็น Child window ที่ Header สร้างขึ้น



Graphics Device Interface

ชื่อนี้นิยมเรียกย่อ ๆ ว่า GDI หมายถึงการอินเตอร์เฟสระหว่างดีไวซ์กับโปรแกรม โดย GDI จะอยู่ใน Software Development Kit ทั้งโปรแกรมและตัววินโดว์สามารถเรียกใช้รูทีนต่าง ๆ ใน GDI ได้สะดวก

Display Context

ในการที่ GDI รูทีนจะติดต่อกับอุปกรณ์ได้นั้นต้องมีที่เก็บข้อมูลเกี่ยวกับดีไวซ์ก่อน โดยเฉพาะอย่างยิ่ง GDI จะต้องมีโครงสร้างข้อมูลที่เก็บรายละเอียดของแต่ละหน้าต่างที่เปิดบนอุปกรณ์เดียวกันหรือแยกจากกัน ซึ่งเราเรียกโครงสร้างข้อมูลนี้ว่าดิสเพลย์คอนเท็กซ์ (D

ay Context) เนื่องจากดิสเพลย์คอนเท็กซ์
นโครงสร้างที่มีขนาดใหญ่มาก และจำ-
นต้องใช้เมื่อต้องการสร้างรูปภาพหรือ
สดงข้อมูลลงบนหน้าต่างเท่านั้น ดังนั้น
มักจะสร้างข้อมูลประเภทนี้เมื่อจะแสดง
อมูลเท่านั้น และจะยกเลิกเมื่อใช้เสร็จ
งตัววินโดว์มีฟังก์ชันในการจัดการ
ยวกับ DC คือ CreateDC และ Re-
oveDC มักใช้คู่กับอีกสองฟังก์ชัน
มอคือ BeginPaint และ EndPaint
ยละเอียดเกี่ยวกับ DC และฟังก์ชัน
กล่าวต่อไปในภายหลัง

DI Output Routines และ Tools

บางส่วนของ GDI Output Rou-
es ได้แสดงไว้ในตารางที่ 3 ซึ่งรูทีน
ล่านี้จะควบคุมตำแหน่งใน client area
อ work space) การเซตสี การแสดง
อความ การลากเส้น การระบายสี
กรอบ และอื่น ๆ

ตารางที่ 3
ตัวอย่าง GDI Output functions

ชื่อรูทีน	OUTPUT
MoveTo	เซตตำแหน่งปัจจุบัน
GetCurrent Position	รับค่าตำแหน่งปัจจุบัน
SetPixel	เซตสีให้ pixel
GetPixel	อ่านค่าสีของ pixel
TextOut	เขียน String
LineTo	
Polyline	
Arc	
Rectangle	
RoundRect	
Polygon	
Pie	
FillRect	
FillRgn	

สิ่งหนึ่งที่เราจำเป็นต้องรู้ไว้ก็คือ GDI output functions ต้องมีการใช้ทูลส์โดยเฉพาะ fonts, pens, brushes อย่างเช่น TextOut ต้องใช้ current font ในการเขียนข้อความ Pie ต้องการปากกา (pen) ในการลากเส้น และต้องการแปรง (brush) ในการระบายสีพื้นที่ภายใน ซึ่งก่อนการใช้งาน GDI ฟังก์ชันจะต้อง Select Object ที่จะใช้ก่อน ทั้ง pen, brush และ font ล้วนแต่มีลักษณะเฉพาะตัวดังต่อไปนี้

– pen จะมี size (line width), style (solid, dashed etc.) และสี

– brush จะมีเหมือน pen แต่ size จะเป็น rectangular แทน

– font จะมี style, height, width ช่องว่างระหว่างบรรทัด ฯลฯ

บทส่งท้าย

ในตอนต่อไปผู้เขียนจะขอกล่าวถึงกราฟิกที่เกี่ยวข้องกับวินโดว์ ได้แก่ Bitmaps, Clipping, Mapping ในส่วนของรายละเอียดอีกครั้งหนึ่งในฉบับต่อไป